

RPGMail Enterprise User Manual (for version 0.01b)

RPGMail Enterprise (RME) User Manual

for RME_v0.01b
2009-12-14



www.MowYourLawn.com

© Copyright 2009, Bartell Software

QUESTIONS ABOUT THIS MANUAL CONTACT aaronbartell@mowyourlawn.com

Table of Contents

Summary.....	2
Features.....	2
Requirements.....	2
Install and Configure.....	3
Start Java Process.....	4
End Java Process.....	5
Examples.....	5
DB2 Tables.....	6
Rebuild and Deploy Java.....	6
Logs and Debugging.....	7

Summary

RPGMail Enterprise (RME) is a follow-on tool to the original RPGMail tool created by Aaron Bartell of www.MowYourLawn.com. Through the eight years of RPGMail's existence there have been a number of updates made but because it is a free and open tool it is hard to justify time to update the tool since it provides no immediate benefit to Aaron Bartell. That is why Aaron has decided to create a "Enterprise" that builds on the feature set of the original RPGMail. Because Aaron is a believer in open source and the ability for a company to have full control of software in their shop, he has determined to keep RPGMail Enterprise open source. This means that when you purchase the Enterprise version you will not only receive the runtime Java and RPG programs but ALSO the complete source code!

Features

- Send email quickly and easily from your RPG program.
- Authentication
- SSL
- Read receipts
- Email priority (i.e. HIGH, NORMAL, LOW)
- Single JVM in batch. This significantly improves performance since there is only ever one JVM running on the machine for RPGMail vs. having each signed on job require one.
- Email log tables so you can see everything that is being sent out from RME.
- Unlimited email attachments
- Unlimited email recipients
- Special ease-of-use API's for those times where you just want a single line of code to send an email (see API RME_simple).

Requirements

RME was created on a AS/400 running V5R3 of OS/400. This version of RME requires at least V5R1 of the operating system. The licensed programs required on your machine are as follows:

5722JV1 Java Developer Kit 1.5 installed.

5722SS1 OS/400 - Portable App Solutions Environment (required by JDK 1.5)

Install and Configure

The following is taken verbatim from the readme.txt file included with the zip file download.

1. Issue the AS/400 command:

```
CRTSAVF FILE(QGPL/RME)
```

2. FTP the file RME from your PC to the AS/400 in BINARY mode

into the save file RME in library QGPL.

3. Issue the AS/400 command:

```
RSTLIB SAVLIB(RME) DEV(*SAVF) SAVF(QGPL/RME)
```

4. Issue the AS/400 commands:

```
MKDIR '/java'  
MKDIR '/java/rme'
```

5. FTP the following files to the AS/400 in binary mode to folder /java/rme/.

Note that you MUST use binary mode when FTPing otherwise you could get unexpected results because there will be an attempted EBCDIC conversion that renders some jars useless.

```
activation.jar  
mail.jar  
jt400.jar  
RPGMailEnt.jar
```

6. Configure mail server.

Update the the "*DFT" record in table RMESVR to have your email server's information.

7. Issue the AS/400 command to start the Java portion that runs in QBATCH:

RPGMail Enterprise User Manual (for version 0.01b)

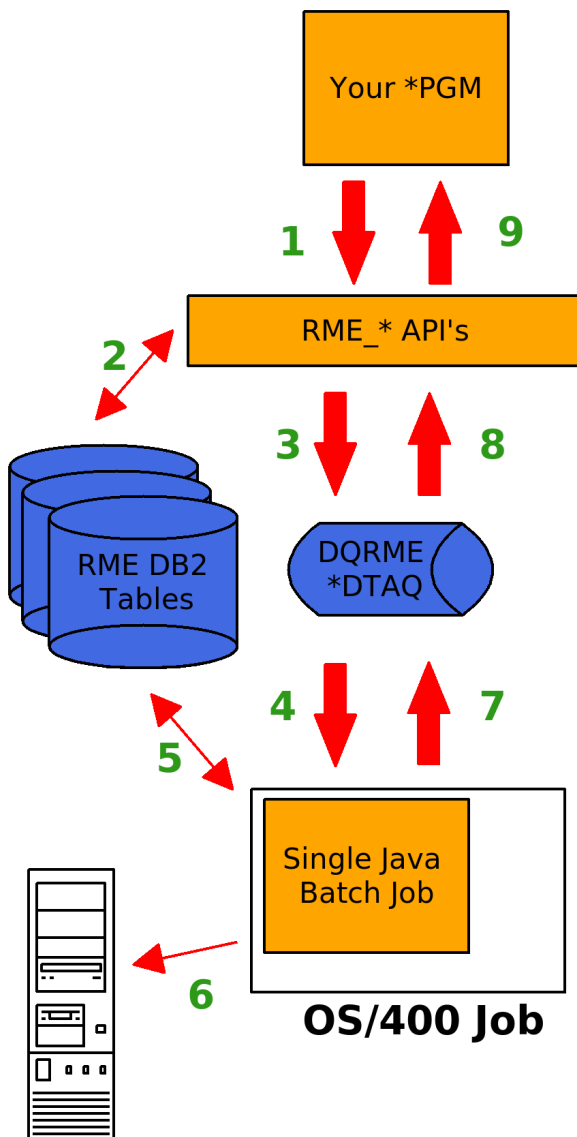
```
ADDLIB RME  
CALL RME/STRMEEZ
```

The following screen shot shows a sample of table RMESVR *DFT entry. If your SMTP server doesn't require authentication then you should specify an F in the Authenticate column.

```
WORK WITH DATA IN A FILE                               Mode . . . . :  CHANGE  
Format . . . . :  RMESVRR                               File . . . . :  RMESVR  
  
Server Id:      *DFT  
Record Created: 2010-01-01-12.00.00.000000  
URI (Domain or IP): smtp.mycompany.com  
-----  
Port:          465  
Authenticate T/F: I  
User Name:     user  
Password:      password
```

Flow of Processing

The following image depicts the flow of information and processing for a typical email being sent using RPGMail Enterprise.



1 - Invoke RME_* API's to provide it with different information about the email (i.e. To/From, Subject, Body, etc).

2 - Each time an RME_* API is called, the values are stored in DB2 tables for later use.

3 - When RME_send() is invoked by your program a record is written to the DQRME data queue for that particular email.

4 - The Java process has been waiting on the data queue this whole time and now receives in the entry that was just written.

5 - The Java process queries the RME DB2 tables to retrieve the email related data.

6 - The Java process uses the JavaMail API's from Sun to make a call to your email server. This can be a local or remote email server.

7 - After the email is sent the Java process will update the status in the RME DB2 tables and respond to the data queue. Note that you can configure it so that your RPG program doesn't wait for a response from the Java process because that could be a needless wait for the user.

8 - The RME_send() API receives the response back from the data queue.

9 - Control is passed back to your program.

Start Java Process

For emails to be sent out there is a Java batch process that needs to be started. Note that the RPG side can be invoked without the Java batch process started and the emails will be retained within the DB2 tables and data queue in a wait state for the Java batch process to start.

There are two methods of starting RME. The first, and more laborious, is invoking the STRRME command which accepts parameters to allow you to alter the various aspects of how it will run. The alternative, and preferred, method is to invoke the STRRMEEZ command which doesn't require any parms. The contents of STRRMEEZ contains a call to STRRME where are of the parms a pre-filled in. This is very useful because entering the Java classpath (CLSPATH parameter) is error prone and tedious. The following can be used to start the Java process using the STRRMEEZ command.

```
ADDLIB RME
CALL RME/STRRMEEZ
```

After starting the Java batch process you should see the following two jobs in your batch subsystem:

Subsystem/Job	User	Type	CPU %	Function	Status
QBASE	QSYS	SBS	.0		DEQW
QJVACMDSRV	AARON	BCI	3.5	PGM-QJVACMDSRV	RUN
RMERUNJVA	AARON	BCH	.0	CMD-RUNJVA	TIMW

End Java Process

When you want to end the Java batch process you can utilize the ENDRME command. The ENDRME command doesn't take any parameters and can be invoked as follows:

```
ADDLIB RME
CALL ENDRME
```

Invoking ENDRME will create a data queue entry that when received by the Java batch process will end it gracefully. Note that you could also end the two Java batch jobs using option 4 in WRKACTJOB and it would be detrimental.

Examples

The example RPG program can be found in source file RME/EXAMPLE,T1_RME. It has also been included here for reference.

```
//-----
// Program.: T_RME
// Author..: Aaron Bartell
// Descr...: Send a simple email using the *DFT RMESVR entry.
// Compile.:
// CRTBNDRPG PGM(RME/T1_RME) SRCFILE(RME/QRPGLESRC) SRCMBR(T1_RME) DBGVIEW(*ALL)
//
//-----
H dftactgrp(*no) bnddir('RMEBND')

/copy qrpglecpy,RME
/copy qrpglecpy>Error

D gUIId          s              15p 0
D gErr           ds              likes(Error_Info) inz
D gProp          ds              likes(RME_Prop) inz
D nl             c              const(x'0D25')
D bdyTxt         s              1024a  varying
/free

monitor;
  gProp.svrId = '*DFT';
  gProp.sbj = 'Is your fridge running?';
  gProp.bdyCntTyp = RME_TEXT;

  gUIId = RME_init(gProp);

  RME_addHdr(gUIId: RME_PrtY: RME_HIGH); // Set as a high priority email

  RME_addAdr(gUIId: RME_FROM: 'myname@company.com': ' ');
  RME_addAdr(gUIId: RME_TO: 'somebody@othercompany.com': ' ');

  bdyTxt = 'My first rpg email app!' + nl +
    ' This is the next line.';
  RME_addToBdy(gUIId: bdyTxt);

  RME_send(gUIId: gErr);
on-error;
  gErr = Error_catch();
endmon;

*inlr = *on;

/end-free
```

DB2 Tables

RMEADR - This table holds all of the email addresses of an email. It is associated to RMEMAIN using the PID column.

RMEATCH - This table holds all of the email attachment locations associated with an email. It is associated to RMEMAIN using the PID column.

RMEBDY - This table holds the email body associated with an email. It is associated to RMEMAIN using the PID column.

RMECFG - This table holds the configuration values that are used by both the Java and RPG processes at runtime.

RMEHDR - This table holds header level name-value-pair information that is associated with an email. It is associated to RMEMAIN using the PID column. This is a generic table that could have any number of email headers applied to it. For example, the "X-Priority" name and value are placed in here to give emails the "High, Normal, Low" priorities that you see in Outlook.

RMELOG - This table holds all of the logs associated with an email. It is associated to RMEMAIN using the PID column. If debugging is turned on in table RMECFG then this table will be written to during the email process.

RMEMAIN - This table is the entity that all other email tables stem from. It contains the status of the email being processed, the server id that is to be used to send the email, the subject, the body content type (i.e. text/plain), and whether or not the RPG process should wait for a response.

RMESVR - This table holds the configuration of an email server that will be used to relay mail traffic from RME. There should always be a record in here with a key value of *DFT which will be the default SMTP server used if none is specified in your RPG business logic program.

Rebuild and Deploy Java

Coming soon!

Logs and Debugging

Logging (aka debugging) is available in RME as a mechanism to do problem determination. Logs are stored primarily in RMELOG by both the RPG and Java processes, but if the Java process isn't able to connect to the IBM i to write records to the log then there will be a file created at the root of the machine that will store debugging information. The format of that file will be */rme_debug_CCYY-Mon-DD.log*.

If you are running into issues with the Java process successfully starting then what I do is start it from Qshell because that allows for Java's standard output to be sent directly to your green screen. The following two steps describe how to do that.

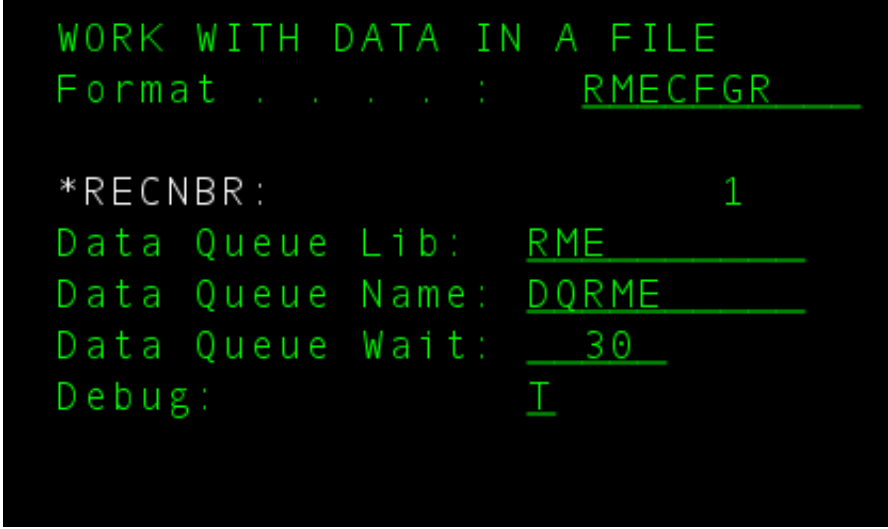
Step 1) Do a STRQSH from the regular green screen command line.

Step 2) Paste the following string onto the Qshell command line and select the <enter> key.

RPGMail Enterprise User Manual (for version 0.01b)

```
/QIBM/ProdData/Java400/jdk15/bin/java -cp  
/java/rme:/java/rme/activation.jar:/java/rme/mail.jar:/java/rme/jt400.jar:/java/rme/RPGMail  
Ent.jar com.mowyourlawn.rme.RPGMailEnt localhost *CURRENT *CURRENT RME
```

By default RME should be configured for debug as shown in the below screen shot. Use a value of T or F (True or False respectively).



```
WORK WITH DATA IN A FILE  
Format . . . . : RMECFGR  
  
*RECNBR: 1  
Data Queue Lib: RME  
Data Queue Name: DQRME  
Data Queue Wait: 30  
Debug: I
```